Artigo SQL Magazine 53 - Replicação Oracle

Mantenha os dados sincronizados para garantir qualidade nos testes de software.

Esse artigo faz parte da revista SQL Magazine edição 53. Clique aqui para ler todos os artigos desta edição

/o:p>

Replicação Oracle

Mantenha os dados sincronizados para garantir qualidade nos testes de software

Amigo leitor, essa matéria é dedicada a você que administra banco de dados Oracle e freqüentemente depara-se com o problema de manter os dados do ambiente de produção de sua empresa atualizados com os demais ambientes existentes como desenvolvimento e homologação.
Essa é uma das mais freqüentes solicitações feitas pela equipe de desenvolvimento e de negócios das corporações. Sem ter em mãos as informações mais atualizadas, fica extremamente difícil realizar os testes para homologar novas versões ou mesmo realizar um suporte adequado para o usuário final.
No outro lado dessa via está a equipe de banco de dados, que se depara com os mais diversos problemas para realizar a sincronização dos dados: a complexidade de se transferir somente os dados sem causar qualquer impacto nos demais ambientes, estruturas não idênticas existentes entre os ambientes, grandes volumes de dados a serem transferidos e a limitação do espaço físico nos ambientes de destino são algumas das situações que contribuem para que a sincronização dos dados não seja feita de forma freqüente.

Planejando a sua replicação
Existem inúmeras formas de se realizar a replicação de dados, partindo da mais simples (como a que será apresentada nessa edição), às mais complexas (a ser

apresentada na próxima edição). Porém, antes de se desenvolver um processo de replicação, é necessário analisar o ambiente e definir qual é o tipo de replicação mais apropriada à necessidade da empresa. Esse planejamento inclui pontos importantes como:

- Identificar os objetos do banco de dados que devem ser replicados;
- Decidir se será necessário criar ambientes com:
  - Replicação única (um único servidor de origem);
  - Multimaster (vários servidores de origem);
  - Visões materializadas no destino;
  - Soluções híbridas citadas acima;
- Decidir qual será a periodicidade de sincronização dos dados;
- Planejar e configurar opções de segurança;
- Configurar a replicação entre os sites;
- Monitorar os ambientes replicados por meio das views do dicionário de dados do Oracle;
- Outros pontos importantes como propagação serial ou paralela, grau de paralelismo, gerenciamento de conflitos e configuração de segurança não devem ser descartados dependendo da criticidade da replicação a ser implementada.

Nesse artigo de replicação utilizando banco de dados Oracle 10g R2, serão utilizadas duas técnicas de replicação.
A primeira técnica apresentada nesta edição é a técnica de replicação manual, destinada a softwares com pequenas quantidades de tabelas, onde é possível analisar o modelo de dados e desenvolver uma sincronização simples e eficiente dos dados, facilitando a implementação em ambientes onde não se têm equipes especialistas destinadas apenas a administrar banco de dados Oracle.
A segunda técnica de replicação, apresentada na próxima edição, será feita pela ferramenta EM (Enterprise Manager) 10g R2 e é destinada a ambientes complexos, críticos e com tratamento para grandes volumes de dados, onde não é possível identificar todos os objetos envolvidos na replicação.

Cenário – Hands On: Projeto Carloca - Venda de software especializado

O projeto Carloca tem como objetivo gerenciar a locação de veículos realizada pelos clientes em determinada loja de veículos automotores.
Para cada nova licença adquirida do software, são instalados dois bancos de dados em servidores diferentes. Um banco de dados é destinado ao ambiente de produção e o outro é o ambiente de homologação, utilizado pela equipe de suporte do Carloca em situações onde existe a necessidade de se analisar alguma anormalidade ou realizar um suporte para suprir determinada necessidade.
Para manter a base de dados de produção sincronizada com o ambiente de homologação, foi desenvolvido um projeto de replicação que garante a atualização dos dados diariamente ou esporadicamente, a partir da execução de um simples job. Nesse sentido, com um estudo realizado concluiu-se que:

- A atualização dos ambientes deverá ser feita diariamente a partir das 22:00 hs;
- O ambiente de homologação irá solicitar ao ambiente de produção a atualização dos dados;
- Qualquer exceção que ocorrer no momento da execução do procedimento deverá ser catalogada em um arquivo de log;

- É necessário analisar o tempo total de execução do procedimento;
- Os dados do ambiente de homologação são utilizados apenas para consulta e atualizações (update).

Após a definição da necessidade do software é necessário analisar o modelo de dados do projeto Carloca, exibido na Figura 1.

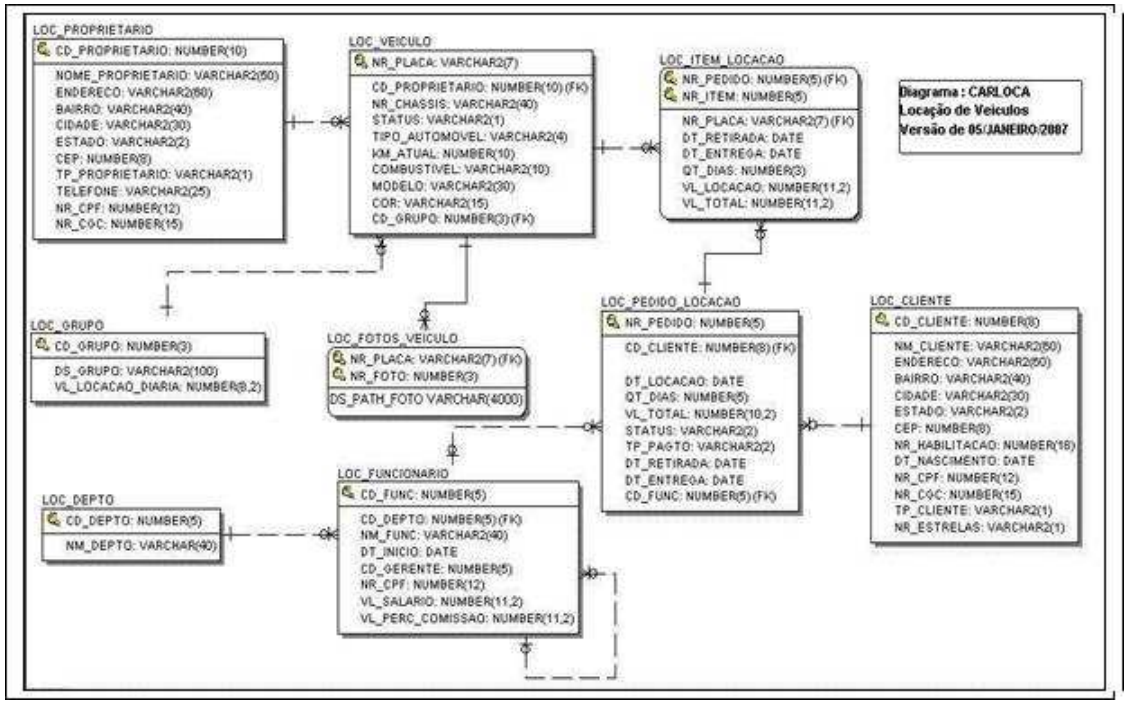

Figura 1. Modelo de dados físico do projeto Carloca.

O modelo de dados do projeto Carloca é composto por apenas nove tabelas, o que facilita a análise e permite sugerir uma simples sincronização de dados a ser feita em ambiente distribuído, exibido na Figura 2.

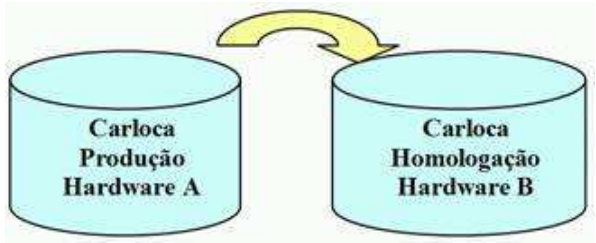

Figura 2. Sincronização de um ambiente distribuído com controle de log.

Solução oferecida

Para resolver a questão de replicação desse projeto, será usada a técnica de replicação manual por meio do comando MERGE.

Isso é possível de se alcançar devido ao conhecimento do modelo de dados do projeto Carloca, exibido anteriormente na Figura 1.

A idéia é fazer a leitura dos dados das tabelas do schema Carloca do ambiente de produção e transferi-las para o ambiente de homologação. Porém, o objetivo é apenas sincronizar os dados de um ambiente de produção para um ambiente de homologação sem ter a preocupação com os demais objetos como: triggers, procedures, functions, packages, constraints, objects type entre outros.

Isso ocorre pelo fato de que todos os objetos passam pela homologação antes de ser aplicado em produção. A premissa para o sucesso dessa implementação depende da igualdade das estruturas (tabelas, colunas e restrições) nos dois ambientes.

Partindo do pressuposto que já existem dois schemas Carlocas instalados e funcionando normalmente, um em cada ambiente (produção e homologação), pode-se dividir as atividades a serem realizadas da seguinte maneira:

- Segurança de acesso no ambiente de produção; criar o usuário Consulta_Carloca;
- Replicar os privilégios de acesso às tabelas para o usuário Consulta_Carloca;
- Criar dblink (característica que permite realizar o acesso entre bancos de dados Oracle distribuídos em outro hardware) de comunicação entre o ambiente de produção e homologação;
- Criar tratamento para o gerenciamento do log de replicação (gerado pelas exceções);
- Desenvolver o procedimento de carga por meio do comando MERGE;
- Criar job para executar o procedimento diariamente às 22:00hs;
- Realizar exaustivos testes para garantir a qualidade do processo.

Mãos à obra

1. Segurança de acesso no ambiente de produção

Como todo ambiente de produção, as informações não podem ficar sujeitas a alterações por usuários indesejados.

Quando se cria um dblink, abre-se uma porta de entrada no ambiente de produção e que deve ser avaliada com extremo rigor. Caso o dblink seja o usuário Carloca (usuário selecionado no dblink para acessar o banco remoto), que é o proprietário das tabelas, é possível realizar comandos DML (insert, update e delete) sem que se realize essa tarefa pela aplicação. Isso quer dizer que um acesso via SQL*Plus ou SQL*Developer pode ser feito, trazendo insegurança em termos de manuseio de informações.

Para resolver essa questão, será criado um usuário chamado Consulta_Carloca no ambiente de produção, e este usuário receberá apenas o privilégio de select nas tabelas do Carloca (Listagem 1).

Listagem 1. Comando que cria o usuário Consulta_Carloca e disponibiliza o privilégio de conexão no banco Oracle.

```
System_PROD>       CREATE USER CONSULTA_CARLOCA
                   IDENTIFIED BY "CST!328"
                   DEFAULT TABLESPACE TSPD_CARLOCA
                  TEMPORARY TABLESPACE TEMP
                   PROFILE DEFAULT
                   ACCOUNT UNLOCK;

User created.

System_PROD>       GRANT CREATE SESSION TO CONSULTA_CARLOCA;

Grant succeeded.
```

2. Replicar os privilégios de acesso às tabelas para o usuário Consulta_Carloca

O usuário Consulta_Carloca foi criado apenas com o privilégio de conexão. Os privilégios das tabelas serão replicados pelo usuário Carloca, proprietário (owner) dos objetos, descrito na Listagem 2.

Listagem 2. Privilégio de acesso (select) ao usuário Consulta_Carloca.

```
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_PROPRIETARIO       TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_CLIENTE            TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_DEPTO              TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_GRUPO              TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_FUNCIONARIO        TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_FOTOS_VEICULO      TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_ITEM_LOCACAO       TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_PEDIDO_LOCACAO     TO CONSULTA_CARLOCA;
Carloca_PROD> GRANT SELECT ON CARLOCA.LOC_VEICULO            TO CONSULTA_CARLOCA;
```

Como parte final do procedimento inicial de configuração, os sinônimos (apelidos) são criados para as tabelas do Carloca. Essa tarefa é feita pelo usuário system ou por algum outro usuário no ambiente de produção que tenha o privilégio adequado (Listagem 3).

Listagem 3. Criando sinônimos (apelidos) para as tabelas do projeto carloca.

```
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_PROPRIETARIO    FOR CARLOCA.LOC_PROPRIETARIO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_CLIENTE       FOR CARLOCA.LOC_CLIENTE;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_DEPTO              FOR CARLOCA.LOC_DEPTO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_GRUPO       FOR CARLOCA.LOC_GRUPO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_FUNCIONARIO    FOR CARLOCA.LOC_FUNCIONARIO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_FOTOS_VEICULO  FOR CARLOCA.LOC_FOTOS_VEICULO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_ITEM_LOCACAO   FOR CARLOCA.LOC_ITEM_LOCACAO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_PEDIDO_LOCACAO FOR CARLOCA.LOC_PEDIDO_LOCACAO;
System_PROD> CREATE SYNONYM CONSULTA_CARLOCA.LOC_VEICULO             FOR CARLOCA.LOC_VEICULO;
```

As tarefas executadas anteriormente garantem que o processo de replicação desenvolvido não causará em hipótese alguma anormalidade em relação aos dados do projeto Carloca.

Esse tipo de implementação é muito importante e relevante em situações de auditoria de processos como SOX (ver Nota 1) ou mesmo trabalhando com modelos de qualidade ISO 9005, CMMI entre outros.

Caminhando nesse sentido, temos até o presente momento um novo usuário criado no ambiente de produção chamado Consulta_Carloca que acessa todas as tabelas da aplicação Carloca.

A próxima etapa será criar uma ligação "link" entre o ambiente de homologação e produção. O usuário responsável pelo acesso aos dados será o Consulta_Carloca.

Nota 1. SOX - Sarbanes-Oxley

Lei americana, assinada em 30 de julho de 2002, pelos senadores Paul Sarbanes (Democrata de Maryland) e Michael Oxley (Republicano de Ohio).

Motivada por escândalos financeiros coorporativos, dentre eles o da Enron e que acabou por afetar drasticamente a empresa de auditoria Arthur Andersen, a Lei foi redigida com o objetivo de evitar o esvaziamento dos investimentos financeiros, e fuga dos investidores, causada pela aparente insegurança a respeito da governança adequada das empresas.

A lei Sarbanes-Oxley, como foi chamada, foi apelidada carinhosamente de Sarbox ou ainda de SOX. Seu conjunto busca garantir a criação de mecanismos de auditoria e segurança confiáveis nas empresas, incluindo ainda regras para a criação de comitês e comissões encarregadas de supervisionar suas atividades e operações de modo a mitigar riscos aos negócios, evitar a ocorrência de fraudes, ou ter meios de identificar quando elas ocorrem, garantindo a transparência na gestão das empresas.

3. Criar dblink de comunicação entre o ambiente de produção e homologação

Para realizar essa tarefa são necessários dois passos.

Realizar a conexão entre os dois servidores

Para realizar essa tarefa, insira o descritor de conexão do servidor de produção no arquivo tnsnames.ora do servidor de homologação, como é exibido na Listagem 4 (geralmente esse arquivo fica localizado no diretório $ORACLE_HOME/network/admin).

Listagem 4. Descritor de conexão do servidor de produção.

```
ORAPROD =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = lnxrh4srvprod)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVER = DEDICATED)
   (SERVICE_NAME = PROD)
  )
 )
```

Para realizar o teste, conecte-se na máquina do banco de dados de homologação com seu respectivo usuário de instalação do Oracle e digite “tnsping oraprod”.
Isso irá fazer com que o servidor de homologação solicite uma resposta do servidor de produção. O retorno deve ser OK (x msec), onde x representa o tempo de retorno de conexão em milisegundos com o servidor de produção.

Criar o *link* (característica *dblink*) de comunicação entre o banco de dados de homologação com o banco de dados de produção e depois realizar um teste de conexão (ver Listagem 5).

Perceba que a partir desse momento a conexão entre os dois ambientes está ativa e operante.
O link privado criado será utilizado pelo usuário Consulta_Carloca no ambiente de homologação e irá acessar as tabelas do usuário Carloca do ambiente de produção.
Perceba que logo após o nome da tabela é utilizado o caractere @(arroba) mais o nome do dblink criado, utilizado para referenciar a outra tabela existente no banco de dados distribuído.

Listagem 5. Criando o link de comunicação entre o ambiente de homologação e produção.

```
Carloca_HOMOLOGACAO> CREATE DATABASE LINK ORAPROD2.WORLD CONNECT TO CONSULTA_CARLOCA IDENTIFIED BY "CST!328" USING 'ORAPROD';

Database link created.
```

```
Carloca_HOMOLOGACAO>  SELECT NOME_PROPRIETARIO FROM LOC_PROPRIETARIO@ORAPROD.WORLD WHERE ROWNUM < 6;

NOME_PROPRIETARIO
-------------------------
Adriana Lopes da Silva
Celso Santos
Yuri Gagari
Maria Linda Oliveira
VeiVei Veiculos Ltda

5 rows selected.
```

Até o presente momento, foram criados os mecanismos para comunicação entre os dois servidores e realizados os devidos testes de conectividade. Nas próximas etapas, são preparados os controles para realizar a transferência dos dados entre as tabelas.

4. Criar tratamento para o gerenciamento do log de replicação

Nessa fase teremos duas tabelas de log, que serão responsáveis por armazenar as execuções realizadas nas replicações entre os ambientes. O modelo de dados apresentado na Figura 3 detalha a sua estrutura.

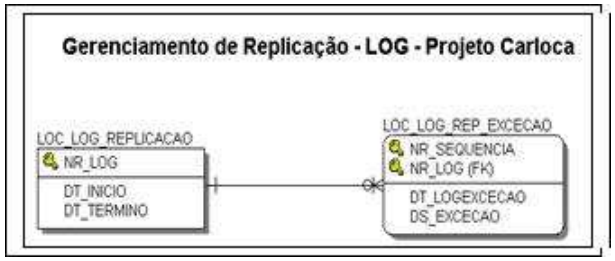

Figura 3. Modelo de dados do controle de gerenciamento de log de replicação.

Para cada replicação executada, será criado um número de log. Qualquer anormalidade encontrada é catalogada na tabela de exceção, permitindo futuras correções. Para armazenar as informações de log, será necessário criar três procedures: prc_log_rep_inicio, prc_log_rep_excecao, prc_log_rep_fim (responsável em finalizar o log).

prc_log_rep_inicio: esse processo é responsável por iniciar o log, ou seja, gerar um número seqüencial que será utilizado por todos os outros processos. A Listagem 6 detalha o código fonte do procedimento. Com o número do log em mãos, é possível durante o processamento armazenar qualquer exceção que ocorra, bem como catalogar o tempo final de processamento do procedimento de replicação.

Listagem 6. Código fonte da procedure de log de início da replicação.

```
Carloca_HOMOLOGACAO> create or replace procedure prc_log_rep_inicio(p_nr_log    out number) is
pragma autonomous_transaction;
begin
begin
                insert into loc_log_replicacao(nr_log, dt_inicio )
        values ( seq_log_replicacao.nextval, current_date) returning nr_log   into p_nr_log;
                commit;
exception
                when others then
        raise_application_error( -20201, 'Erro Crítico * Insert log replicação ' || SqlErrM);
end;
end prc_log_rep_inicio;
/
Procedure created.
```

prc_log_rep_excecao: esse processo é responsável por armazenar qualquer exceção ocorrida durante o processamento da sincronização dos dados (por exemplo, no momento da carga ocorreu uma conversão inválida de valores ou datas; diferentes tipos de dados ocorridos por falta de estruturas de idênticas nos outros ambientes, entre outras). Essa procedure deve ser embutida dentro de cada procedure de carga, para que seja possível identificar o ponto exato onde ocorreu o problema. O código fonte desse procedimento é detalhado na Listagem 7.

Listagem 7. Código fonte da procedure de log de exceções da replicação.

```
Carloca_HOMOLOGACAO> create or replace procedure prc_log_rep_excecao(
p_nr_log        in number,
p_ds_excecao        in varchar2
) is
pragma autonomous_transaction;
v_nr_sequencia   loc_log_rep_excecao.nr_sequencia%type;
begin
```

```
        select nvl(max(nr_sequencia),0) + 1 into v_nr_sequencia
        from  loc_log_rep_excecao where nr_log = p_nr_log;

begin
        insert into loc_log_rep_excecao(nr_log,nr_sequencia, ds_excecao, dt_logexcecao)  values (p_nr_log, v_nr_sequencia, p_ds_excecao, current_date);
          commit;
exception
        when others then
        raise_application_error( -20202, 'Erro Crítico * Insert log replicação ' || SqlErrM);
end;
end prc_log_rep_excecao;
/

Procedure created.
```

prc_log_rep_fim: esse processo é responsável em finalizar a execução do processo de replicação, informando o horário final em timestamp do processamento. A Listagem 8 detalha o código fonte do procedimento.

Listagem 8. Código fonte da procedure de log de finalização da replicação.

```
Carloca_HOMOLOGACAO> create or replace procedure prc_log_rep_fim (
p_nr_log    in number ) is
pragma autonomous_transaction;
begin
begin
        update loc_log_replicacao set dt_termino = current_date where  nr_log = p_nr_log;
          commit;
exception
                when others then
        raise_application_error( -20201, 'Erro Crítico * Finaliza log  replicação' || SqlErrM);
end;
end prc_log_rep_fim;
```

```
/

Procedure created.
```

Com as tabelas e os procedimentos de log em mãos, a próxima etapa é desenvolver o script de replicação de dados.

5. Desenvolver o procedimento de carga por meio do comando MERGE

Nesta fase, é possível utilizar inúmeras técnicas para realizar a replicação dos dados. É possível desabilitar as constraints existentes, fazer a carga dos dados e posteriormente habitá-las novamente, ou até mesmo montar um procedimento de leitura das dependências dos objetos pelo dicionário do banco de dados e desenvolver a seqüência de execução da replicação.

A idéia aqui é ser simplista. Partindo do pressuposto que é necessário conhecer o modelo do banco de dados, vamos montar a replicação das tabelas em procedures isoladas e depois montar um procedimento único que irá determinar a seqüência de execução dos procedimentos, bem como ativar o log de replicação.

Veja o desenvolvimento dos procedimentos de carga nas Listagens 9 e 10.

Listagem 9. Código fonte da procedure de carga da tabela de cliente do projeto Carloca.

```
Carloca_HOMOLOGACAO> create or replace procedure prc_replicacao_loc_cliente( p_nr_log in number) is
begin
	merge into loc_cliente c
		using loc_cliente@oraprod.world cprod
		on (c.cd_cliente = cprod.cd_cliente)
	when matched then
		update set
			c.nm_cliente		= cprod.nm_cliente,
			c.endereco		= cprod.endereco,
			c.bairro		= cprod.bairro,
			c.cidade		= cprod.cidade,
			c.estado		= cprod.estado,
			c.cep			= cprod.cep,
			c.nr_habilitacao	= cprod.nr_habilitacao,
			c.dt_nascimento		= cprod.dt_nascimento,
			c.nr_cpf		= cprod.nr_cpf,
			c.nr_cgc		= cprod.nr_cgc,
```

```
                c.tp_cliente          = cprod.tp_cliente,
                c.nr_estrelas         = cprod.nr_estrelas
        when not matched then
                insert (c.cd_cliente, c.nm_cliente, c.endereco, c.bairro, c.cidade,c.estado,c.cep, c.nr_habilitacao,
c.dt_nascimento,c.nr_cpf,c.nr_cgc,c.tp_cliente,c.nr_estrelas)values (cprod.cd_cliente, cprod.nm_cliente, cprod.endereco,cprod.bairro,cprod.cidade,
        cprod.estado, cprod.cep, cprod.nr_habilitacao, cprod.dt_nascimento, cprod.nr_cpf,
                cprod.nr_cgc,cprod.tp_cliente,cprod.nr_estrelas);
exception
        when others then
                 prc_log_rep_excecao(p_nr_log, SQLErrM);
end prc_replicacao_loc_cliente;
/
Procedure created.
```

A técnica utilizada na Listagem 9 é simples: saindo do ambiente de homologação, vá ao banco de dados de produção pelo usuário Consulta_Carloca, faça uma junção com a tabela de clientes do usuário Carloca(produção) e recupere todos os dados para o usuário Carloca(homologação). Caso a restrição pela chave primária seja encontrada (join), atualize todas as colunas com exceção da PK (primary key), caso não encontre informações identificadas pela junção, realize uma inserção dos dados.

A Listagem 10 apresenta a implementação para a tabela Pedido de Locação.

Listagem 10. Código fonte da procedure de carga da tabela de pedidos de locação do projeto Carloca.

```
Carloca_HOMOLOGACAO> create or replace procedure prc_replicacao_loc_pedido( p_nr_log in number) is
begin
        merge into loc_pedido_locacao p
                using loc_pedido_locacao@oraprod.world pprod
                on (p.nr_pedido = pprod.nr_pedido)
        when matched then
                update set
                p.cd_cliente  = pprod.cd_cliente,
                p.dt_locacao  = pprod.dt_locacao,
                p.qt_dias     = pprod.qt_dias,
                p.vl_total    = pprod.vl_total,
```

```
                p.status      = pprod.status,
                p.tp_pagto    = pprod.tp_pagto,
                p.dt_retirada = pprod.dt_retirada,
                p.dt_entrega  = pprod.dt_entrega,
                p.cd_func     = pprod.cd_func
        when not matched then
                insert values(pprod.nr_pedido,pprod.cd_cliente,pprod.dt_locacao,pprod.qt_dias,pprod.vl_total,
                pprod.status,pprod.tp_pagto,pprod.dt_retirada,pprod.dt_entrega,pprod.cd_func);
exception
        when others then
                prc_log_rep_excecao(p_nr_log, SQLErrM);
end prc_replicacao_loc_pedido;
/
Procedure created.
```

O código da Listagem 10 segue a mesma diretriz detalhada na Listagem 9. Assim sendo, para cada tabela envolvida, deve-se criar os seus respectivos procedimentos de carga. Nesse momento não é importante saber a seqüência de execução, que será tratada agora pela procedure prc_replicacao_carloca, conforme detalhado na Listagem 11.

Listagem 11. Código fonte da procedure principal de replicação do projeto Carloca.

```
Carloca_HOMOLOGACAO> create or replace procedure prc_replicacao_carloca is
v_nr_log        loc_log_replicacao.nr_log%type;
begin
prc_log_rep_inicio( v_nr_log );  -- irá receber o número do log da replicação e iniciar o tempo de execução
prc_replicacao_loc_proprietari( v_nr_log);
prc_replicacao_loc_cliente( v_nr_log);
prc_replicacao_loc_grupo( v_nr_log);
prc_replicacao_loc_depto( v_nr_log);
prc_replicacao_loc_funcionario( v_nr_log);
prc_replicacao_loc_veiculo( v_nr_log);
prc_replicacao_loc_fotos_veic( v_nr_log);
```

```
prc_replicacao_loc_pedido( v_nr_log);
prc_replicacao_loc_item_pedido( v_nr_log);
commit;
prc_log_rep_fim( v_nr_log );     -- irá finalizar atualizando o tempo de execução do processo
end prc_replicacao_carloca;
/
Procedure created.
```

A procedure prc_replicacao_carloca tem a finalidade de agrupar todas as procedures que irão fazer carga nas tabelas do projeto Carloca, fazendo com que sua simples execução dispare todas as cargas de replicação de dados. Essa procedure necessariamente tem que seguir uma seqüência lógica na sua execução, pois as restrições de chave estrangeira (FK) estão habilitadas e serão ativadas caso encontre alguma anormalidade. Uma opção um pouco mais trabalhosa, porém eficiente, seria obter os relacionamentos das tabelas por meio da leitura das visões do banco de dados.

6. Criar Job para executar o procedimento diariamente às 22:00hs

Nesta etapa, um job manual é criado utilizando-se a ferramenta SQL*Plus. Esse job tem a finalidade de agendar a execução diária no horário noturno (a partir das 22:00hs) do processo de replicação dos dados do projeto Carloca. Conectado como usuário Carloca, no ambiente de homologação, execute os comandos apresentados na Listagem 12.

Listagem 12. Criando um Job por meio da ferramenta SQL*Plus para executar a replicação diariamente.

```
Carloca_HOMOLOGACAO> variable njob number

Carloca_HOMOLOGACAO> execute  dbms_job.submit(job=>:njob, what=>'prc_replicacao_carloca;'   , next_date=>trunc(sysdate)+22/24,
interval=>'trunc(sysdate+1)+22/24');
Carloca_HOMOLOGACAO> commit;
Commit complete.
```

O job vai executar a procedure prc_replicacao_carloca que é composta por todas as outras procedures que fazem as cargas nas tabelas do projeto carloca. O horário de execução estabelecido é 22:00hs com uma freqüência diária. Apesar desse job ter sido criado manualmente dentro da ferramenta SQL*Plus, ele será gerenciado automaticamente pelo banco de dados Oracle.

7. Realizar exaustivos testes para garantir a qualidade do processo

Para finalizar nossa replicação manual, será cadastrado um novo Cliente e também será cadastrado um pedido de locação para dois automóveis. A Listagem 13

mostra os comandos utilizados.

Listagem 13. Inserindo valores na tabela de produção para realizar os testes necessários.

```
-- Inserindo um cliente
insert into loc_cliente
(cd_cliente, nm_cliente, endereco, bairro, cidade, estado, cep, nr_habilitacao, dt_nascimento,
nr_cpf, nr_cgc, tp_cliente, nr_estrelas)
values ( 1255, 'Gustavo Lopes Dedón', 'Av. Brigadeiro Luis Antonio, 1140', 'Bela Vista', 'São Paulo', 'SP' ,          01317000, 035385258, to_date('18/12/1980','dd
/mm/yyyy'), 09308388877, null, 'F', 0);
1 row created.

-- Inserindo uma locação
insert into loc_pedido_locacao
(nr_pedido, cd_cliente, dt_locacao, qt_dias, vl_total, status, tp_pagto, dt_retirada, dt_entrega, cd_func) values
(241, 1255, sysdate, 4, 600, 'PG', 'CC', trunc(sysdate)+1, trunc(sysdate)+5, 5);
1 row created.

-- Insert com 2 linhas na tabela item locacao
insert all
into loc_item_locacao
(nr_pedido,nr_item,nr_placa,dt_retirada,dt_entrega,qt_dias,vl_locacao,vl_total) values (241, 1,'CZR9988',  trunc(sysdate)+1, trunc(sysdate)+5, 4,   100, 400)
into loc_item_locacao
(nr_pedido,nr_item,nr_placa,dt_retirada,dt_entrega,qt_dias,vl_locacao,vl_total) values (241, 2,'DRV4395',  trunc(sysdate)+1, trunc(sysdate)+5, 4,   50, 200)
select * from dual;
2 rows created.

Commit;

Commit complete.
```

Dirigindo-se para a parte final do projeto de replicação, as informações ainda não foram atualizadas, pois o job que está agendado ainda não entrou em ação.
Para que as informações sejam replicadas a qualquer momento, a procedure principal prc_replicacao_carloca será executada manualmente, forçando a sincronização

dos dados.

A Figura 4 mostra essa execução, exibindo as informações antes e depois da execução da procedure de replicação.

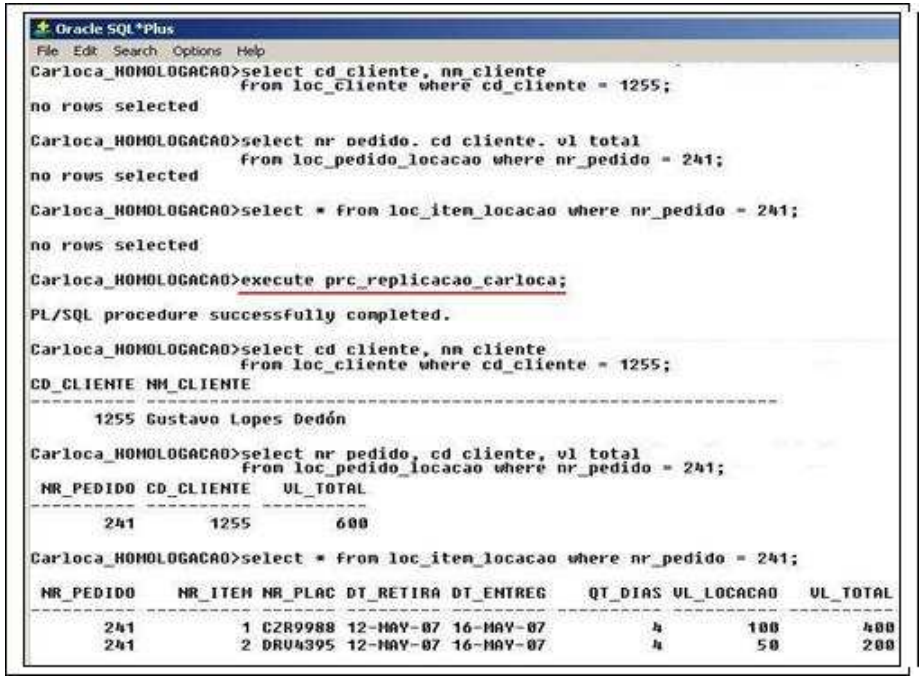

Figura 4. Execução da procedure de replicação para que os dados estejam sincronizados entre o ambiente de Homologação e Produção.

Conclusão

A técnica utilizada nesse artigo é uma idéia simples de replicação de dados para pequenos softwares e com pouco volume de dados.

Outras técnicas consagradas como: export / import de dados, limpeza dos dados das tabelas antes das cargas, desabilitar constraints, eliminar os índices antes da carga dos dados e depois recriá-los, entre outras aceleram e qualificam seu projeto de replicação, pois o objetivo a ser alcançado é disponibilizar as informações nos ambientes de testes o mais próximo possível ao de produção, no menor tempo possível. O sucesso da replicação está associado a uma técnica adequada de testes, para garantir o resultado planejado.

O sincronismo permite à equipe de desenvolvimento e homologação de novas versões de softwares realizar simulações próximas às encontradas no ambiente de produção.

Na próxima edição, as técnicas avançadas de replicação por meio da ferramenta gráfica EM 10g R2 serão implementadas para cenários de ambientes críticos, onde o tempo pode fazer diferença entre a prosperidade e a ruína.